A UML-BASED APPROACH TO THE MODELLING AND SUPERVISORY CONTROL OF MANUFACTURING MACHINERY

B. Bordbar, L. Giacomini and D.J. Holding[1]

1. Introduction

The paper addresses the problem of designing the co-ordination logic for the supervisory control of multi-component systems, such as manufacturing machinery. The main objective is to take data describing the behaviour of each type of component and the "rules" that govern the co-ordination and synchronisation of the components and to synthesise appropriate coordination logic and a verifiable model of the system. The premise is that the machine/plant operation can be split into a set of phases or motions, with continuous regulatory control through each of the phases and logical decision, which determine the next phase or motion, computed at the end of each phase. Our approach combines a structured methodology based on UML (Unified Modelling Language) [1,3] with qualitative techniques that provides reasoning and validation tools.

Based on UML design procedure, we start by analysing the UML Use Cases, which are detailed description of the system objectives and component behaviours. This enables us to construct abstract continuous and discrete event models of the components and the synchronisation logic. In UML the dynamics of objects are described using a form of state diagram known as a Statechart [4]. Although Statecharts are very popular and are well supported by implementation tools, they currently lack analytic capabilities and thus software tools cannot ensure the functional consistency of the overall design. To facilitate reasoning about the asynchronous concurrent behaviour of the components we replace the Statechart by an analytic Petri net [2,5] model which can be analysed using Petri net theory. A novel algorithm is provided to synthesis the synchronisation logic and create a Petri net model of the composite controlled system. Exploring the behaviour of this Petri net helps in verification of the safety of the plant.

Our approach is explained by modelling of a typical wrapping machine.

2. UML based design

Consider the design of a controller for a simplified production line comprising loosely-coupled independently-driven mechanisms such as conveyor belts, wrapping film feeders, film sealers and cutters as shown in Fig. 1. The major modules of the system are controlled individually and independently and perform motion profiles corresponding to different tasks. Supervisory (discrete event) control is to be used to synchronise the different parts together.

2.1 Use Case

The UML design procedure [1] starts with the study of the Use Cases which are detailed written descriptions of ‘what the objectives are’ and 'how the job is carried out'. Studying the use cases enables the designer to recognise different '*key agents*' of the system, known as *Objects* in UML terminology. For example, the wrapping system of Fig. 1 is made of 4 objects: the belt, the foil roll unwinding device (film), the welder, the cutter. The product (JOB) and the foil that carries a printed tag (TAG) are identified with their supports, i.e. the belt and the film, respectively. JOB and TAG are displaced with respect to the belt and film.

Let us examine the Belt Use Cases. When the JOB arrives (*new* JOB) in the proximity of a decision point sensor (**dp**), the state of the TAG is evaluated. If the TAG is at decision point the wrapping can take place (*go*). However, if the TAG is still **out**side the wrapping area, the JOB will stop, **wa**iting for the TAG to arrive at its decision point (*ab*ort operation). When the TAG arrives (*new* TAG), the JOB is restarted (*st*art leading to the **wr**apping state). When JOB and TAG are both in the **wr**apping state, the packaging foil is formed into a tube via a funnel, and a longitudinal sealing roller welds the two edges of the film together. The tube is sealed between packs by a lateral sealer (welder) and the wrapped product exits from the **wr**apping area (*ex*it leading to the state **out**). The sealed products are then separated by a cutting machine (cutter) to produce individually packaged products, and the whole cycle restarts.

Similar behavioral models have been derived for the Film (TAG), Welder and Cutter. The welder and film, and film and cutter, are synchronised by applying a heuristic similar to the one between the belt and film.

2.2 Class diagrams

Considering common features and operations of key agents, objects are extrapolated into collections called

[1] Dept. of Electronic Engineering, Aston University, Aston Triangle, Birmingham B4 7ET, UK {b.bordbar, l.giacomini, d.j.holding}@aston.ac.uk

Classes. Each class has a set of attributes representing the state of an object and a set of operations representing the actions that the class can perform. In the description in Section 2.1 (Use Case for the production line of Fig. 1), the underlined terms represent the classes. The product to be wrapped, JOB, is identified with the belt. The printed film and the motor driving the unwinding are identified with the Film object. The terms in bold typeface are the attributes of the classes (for the Belt, B_dp, B_wait, B_wrap, B_out; F_dp and similarly for the class Film). The terms in italic typeface are the operations of the class.

Classes can be organised in a graph (or a collection of graphs), to build a '*class diagram*', that describes the *static relationship* between the classes.

3 Dynamic Model

The UML dynamic model describes behavioural aspects of the object classes. In this paper a Petri net replaces the UML dynamic model (for general information regarding Petri nets, we refer to [5]).

First we derive Petri nets for each of the classes by assigning one place to each attribute and one transition to each operation. Places associated with attributes that an operation sets to False (or True) form inputs (or outputs) of the associated transition. In this particular application, the dynamic models of the classes are all structured as shown in Fig. 2. We create an initial marking for each instantiated object by considering the initial state of the corresponding components of the production line.

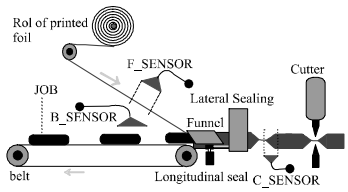

Fig. 1 : Production Line

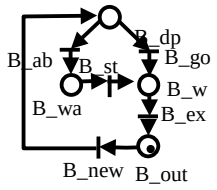

Fig. 2: PN for the class Belt

4 Precedence relationships and synchronisation.

To maintain the precedence relationships and the synchronisation objectives specified in the Use Case we use the Use Case information to directly construct a directed graph which enumerates all desirable states and their relationships. The resulting *Graph of Desirable States* (GDS) embraces all we expect the system to do, and identifies any unwanted or undesirable behaviour which must be prohibited.

Assume that a system is made of m objects, and that the dynamic model of each object is a Petri net that will form a component of the composite design. Typically, the objects are synchronised two at a time, until the compositional approach encompasses the whole system.

For conciseness, we will focus on the interaction and synchronisation of the Belt and Film.

Let the component Petri nets be bounded and live and represented by $(\boldsymbol{N}_B, \mathbf{m}^B{}_0)$, $(\boldsymbol{N}_F, \mathbf{m}^F{}_0)$, where $\mathbf{m}^B{}_0$, $\mathbf{m}^F{}_0$ denote the initial markings of Belt and Film. Let $R_\infty(\boldsymbol{N}_i, \mathbf{m}^i{}_0)$, denotes the set of all reachable markings of the Petri Net $(\boldsymbol{N}_i, \mathbf{m}^i{}_0)$, i = B,F. For each $\mathbf{m}^i \in R_\infty(\boldsymbol{N}_i, \mathbf{m}^i{}_0)$, let *enabled*$(\mathbf{m}^i)$ denote the set of all enabled transitions of $\boldsymbol{N}_i$ under the marking $\mathbf{m}^i$. Each node of GDS is labelled by a 3-tuple of the form $a = (\mathbf{m}^B, \mathbf{m}^F, U)$ where $\mathbf{m}^B$, $\mathbf{m}^F$ are reachable markings of the components $\boldsymbol{N}_B$, $\boldsymbol{N}_F$ and U is the set (possibly empty) of undesirable enabled transitions under $\mathbf{m}^B$, $\mathbf{m}^F$, as derived from the use case. Thus U is a subset of *enabled*$(\mathbf{m}^B) \cup$ *enabled*$(\mathbf{m}^F)$. For the node labelled with $a = (\mathbf{m}^B, \mathbf{m}^F, U)$ we shall write $\boldsymbol{m}(a) = (\mathbf{m}^B, \mathbf{m}^F)$ and $U(a) = U$.

The GDS can be generated as follows:

Step1: Create the first node, which shall be referred to as the *initial node*, and label it with $a_0 = (\mathbf{m}^B{}_0, \mathbf{m}^F{}_0, U_0)$ where U_0 is the (possibly empty) subset of enabled transitions which are undesirable.

Step2: While there is a node a that *Step2* is not applied to, do the following:

Let $a = (\mathbf{m}^B{}_a, \mathbf{m}^F{}_a, U_a)$. For each $t \in$ *enabled*$(\mathbf{m}^B{}_a)$, where t does not belong to U_a create a "*child*" node $b = b_t = (\mathbf{m}^B{}_b, \mathbf{m}^F{}_a, U_b)$, where $\mathbf{m}^B{}_a [\boldsymbol{N}_B, t\rangle \mathbf{m}^B{}_b$ and, according to the use case (see Section 2.1), U_b is the set of enabled transitions under $\mathbf{m}^B{}_b$, $\mathbf{m}^F{}_a$ that are to be prevented from firing. Then the node a is connected to each child b_t via an edge which is labelled by t. To preserve the uniqueness of the labelling of nodes, for each child b_t if there is another node c with the same labelling as b_t, then b_t is cancelled and edge input to b_t is diverted into c. Analogous procedure applies for each $t \in$ *enabled*$(\mathbf{m}^F{}_a)$.

For example, at state (B_out, F_dp), the set of enabled transitions {B_new, F_ab, F_go}. Because of the Use Case, if the TAG is at decision point but the JOB is still out of scope, then we want to decelerate the film, until complete rest if needed. Thus the transition F_go is undesirable: U = {F_go}. Proceeding in this way the graph in Fig. 3 is built.

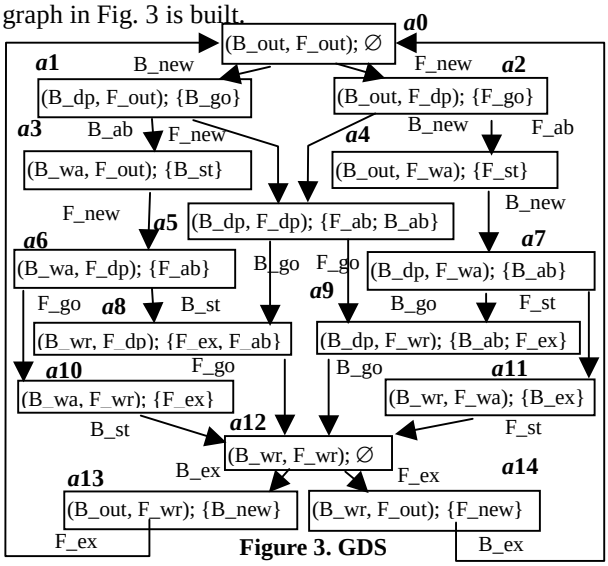

Figure 3. GDS

The resulting GDS condenses, in a Petri Net style, all the information about the dynamics of the two co-operating

subsystems. The GDS can reveal problems with a design: for example, if there is a node *a* of GDS with no output then our design of the system expects a deadlock, which is anomalous.

5 Connecting the components Petri nets

Consider the task of interconnecting the dynamic models, to create a new composite system with the desired co-ordination and synchronisation as described in the Use Cases. The composition is performed using the information in the GDS concerning desirable and undesirable transitions.

Synthesis Method: To ensure the correct evolution of the composite net it is necessary to create a critical section for the synchronisation logic associated with those nodes that have non-empty sets of undesirable transitions. More formally, for each $t \in \Delta$ (Δ denotes the union of all undesirable transitions $U(a)$) create a new place $s(t)$ as an input to t and for each $a \in \{ a \mid t \in D(a)\}$, where $D(a)$, the set of all enabled transitions that are not undesirable, create a new transition proxy(t ; *a*) as an input transition to $s(t)$. Connect all places marked under $\boldsymbol{m}(a)$ to proxy(t; *a*) with double direction arrow. To control the associated critical section, create a place *CtrlSec*(*a*) occupied by a token. Connect *CtrlSec*(*a*) as an input to all proxy(t; *a*) and connect $t \in D(a)$ as an input to *CtrlSec*(*a*). For each $t \in \Delta$, if there are $a \neq b$ such that t is input to both *CtrlSec*(*a*) and *CtrlSec*(*b*), then cancel *a* and divert all inputs (output) transitions of *a* which are not input (output) to *b* into inputs (outputs) of *b*, respectively.

Applying the above procedure to the Petri Nets of Belt and Film, i.e. component Petri net ($\boldsymbol{N}_B$, $\mathbf{m}^B{}_0$), ($\boldsymbol{N}_F$, $\mathbf{m}^F{}_0$), and using the GDS in Fig. 3, we generate a rather complex composite Petri Net, i.e. Petri net ($\boldsymbol{N}$, $\mathbf{m}_0$), which performing exactly the tasks required. From that design, standard net reduction techniques have been used to simplify the net without changing its functionality. The result of this procedure is the Petri Net in Fig. 4.

6 Evaluation

The Petri Net in Fig. 4, is deadlock free and bounded. Its reachability graph has 15 states which match the states in the GDS, is straightforward to verify that the synthetised Petri Net does what required. In general, such verificationis straightforward using a model checker such as DesignCPN. The design in figure 4, has been used to design the control strategy for a Matlab (vers. 5.3) model and the outcome of a simulink experiment is shown in Fig. 5, which demonstrates correct functional behavior.

7 Conclusions

This paper has presented an integrated approach to UML for modelling and analysing discrete event controllers for real-time manufacturing systems. It has shown that Petri-net theory can be used to improve the representation and analysis of the dynamic model of such systems, making the design engineer more confident that the model accurately represents the system. Also, it has shown that UML Use Case information and compositional Petri net techniques can be used to design the co-ordination and synchronisation logic for such systems. The methods and algorithms presented in the paper facilitate the automatic design of the synchronisation logic, and open the possibility of scalable designs for large scale or compositional systems. The composite Petri-net model synthesised using these techniques can be used to implement a controller based on supervisory control theory.

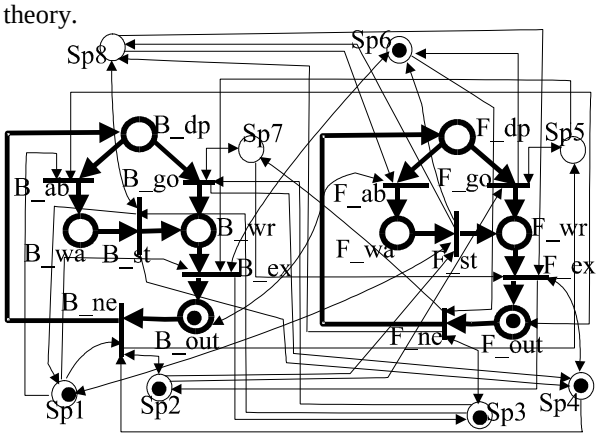

Figure 4: Petri Net for the synchronisation of Belt and Film.

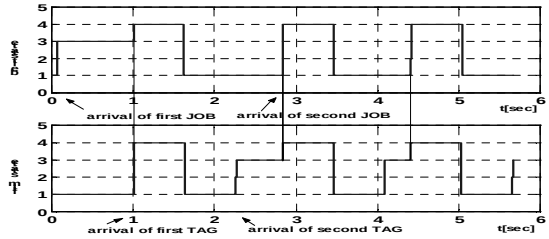

Figure 5: States of Film and Belt in the chart: 1=_out, 2=_dp, 3=_wa, 4=_wr.

Acknowledgements

This work was supported by EPSRC (UK) Grant GR/L31234.

References

1. Booch, G., J. Rumbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley, 1999.

2. Desrochers, A.A. and R.Y. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems*, IEEE Press, 1995.

3. Douglass, B.P., *Doing Hard Time. Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*, Addison Wesley, 1999.

4. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, Vol. 8, pp. 231--274, 1987.

5. Murata, T., Petri Nets: properties, analysis and applications, *Proceedings of the IEEE*, vol. 77, No.4, pp. 541-580, 1989.